Title of Discipline:

Computer-Aided Analysis of Electronic Circuits

Laboratory Lecture 9

Bachelor : Telecommunication Technologies and Systems

Year of Study: 2

Computer-Aided Analysis of Electronic Circuits

Laboratory 9
PSpice Multi-run analyses and device modeling

Multi-run commands in PSpice

.STEP (parametric analysis)

.TEMP (temperature)

Device modeling commands in PSpice

.MODEL (model definition)

.SUBCKT (subcircuit)

.ENDS (end subcircuit)

.DISTRIBUTION (user-defined distribution)

.STEP command

Purpose

The .STEP command performs a parametric sweep for all of the analyses of the circuit.

The .STEP command is similar to the .TEMP (temperature) command in that all of the typical analyses—such as .DC (DC analysis), .AC (AC analysis), and .TRAN (transient analysis)— are performed for each step.

Once all the runs finish, the specified .PRINT (Print) table or .PLOT (Plot) plot for each value of the sweep is an output, just as for the .TEMP or .MC (Monte Carlo Analysis) command.

Probe displays nested sweeps as a family of curves.

General Form

.STEP LIN <sweep variable name> <start value> <end value> <increment value>

.STEP [DEC |OCT] <sweep variable name> <start value> <end value> <points value>

.STEP <sweep variable name> LIST <value>*

The first general form is for doing a linear sweep. The second form is for doing a logarithmic sweep. The third form is for using a list of values for the sweep variable.

.STEP command

Examples

```
.STEP VCE 0V 10V .5V
.STEP LIN I2 5mA -2mA 0.1mA
.STEP RES RMOD(R) 0.9 1.1 .001
.STEP DEC NPN QFAST(IS) 1E-18 1E-14 5
.STEP TEMP LIST 0 20 27 50 80 100
.STEP PARAM CenterFreq 9.5kHz 10.5kHz 50Hz
```

The first three examples are for doing a linear sweep. The fourth example is for doing a logarithmic sweep. The fifth example is for using a list of values for the sweep variable.

Arguments and options

Sweep type

The sweep can be linear, logarithmic, or a list of values. For [linear sweep type], the keyword LIN is optional, but either OCT or DEC must be specified for the <logarithmic sweep type>.

.STEP command

The sweep types are described below.

Sweep types	Meaning
LIN	Linear sweep. The sweep variable is swept linearly from the starting to the ending value. The <increment value> is the step size
DEC	Sweep by octaves. The sweep variable is swept logarithmically by octaves. The <points value> is the number of steps per octave.
OCT	Sweep by decades. The sweep variable is swept logarithmically by decades. The <points value> is the number of steps per decade.
LIST	Use a list of values. In this case there are no start and end values. Instead, the numbers that follow the keyword LIST are the values that the sweep variable is set to.

.STEP command

<sweep variable name>

The <sweep variable name> can be one of the types described below.

Sweep variable name	Meaning
Source	A name of an independent voltage or current source. During the sweep, the source’s voltage or current is set to the sweep Value.
Model parameter	A model type and model name followed by a model parameter name in parenthesis. The parameter in the model is set to the sweep value.
Temperature	Use the keyword TEMP for <sweep variable name>. The temperature is set to the sweep value. For each value in the sweep, all the circuit components have their model parameters updated to that temperature.
Global parameter	Use the keyword PARAM, followed by the parameter name, for <sweep variable name>). During the sweep, the global parameter’s value is set to the sweep value and all expressions are reevaluated.

.STEP command

<start value>

Can be greater or less than <end value>: that is, the sweep can go in either direction.

<increment value> and <points value>

Must be greater than zero.

Comments

The .STEP command is similar to the .DC (DC analysis) command and immediately raises the question of what happens if both .STEP and .DC try to set the same value. The same question can come up using .MC (Monte Carlo analysis). The answer is that this is not allowed: no two analyses (.STEP, .TEMP (temperature), .MC, .WCASE (sensitivity/worst-case analysis), and .DC) can try to set the same value. This is flagged as an error during read-in and no analyses are performed.

You can use the .STEP command to look at the response of a circuit as a parameter varies, for example, how the center frequency of a filter shifts as a capacitor varies. By using .STEP, that capacitor can be varied, producing a family of AC waveforms showing the variation. Another use is for propagation delay in transient analysis.

.STEP command

<start value>

Can be greater or less than <end value>: that is, the sweep can go in either direction.

<increment value> and <points value>

Must be greater than zero.

Comments

The .STEP command is similar to the .DC (DC analysis) command and immediately raises the question of what happens if both .STEP and .DC try to set the same value. The same question can come up using .MC (Monte Carlo analysis). The answer is that this is not allowed: no two analyses (.STEP, .TEMP (temperature), .MC, .WCASE (sensitivity/worst-case analysis), and .DC) can try to set the same value. This is flagged as an error during read-in and no analyses are performed.

You can use the .STEP command to look at the response of a circuit as a parameter varies, for example, how the center frequency of a filter shifts as a capacitor varies. By using .STEP, that capacitor can be varied, producing a family of AC waveforms showing the variation. Another use is for propagation delay in transient analysis.

.STEP command

Usage examples

1) The .STEP command only steps the DC component of an AC source. In order to step the AC component of an AC source, a variable parameter has to be created. For example,

```
Vac 1 0 AC {variable}
.param variable=0
.step param variable 0 5 1
.ac dec 100 1000 1e6
```

2) This is one way of stepping a resistor from 30 to 50 ohms in steps of 5 ohms, using a global parameter:

```
.PARAM RVAL = 1
R1 1 2 {RVAL}
.STEP PARAM RVAL 30,50,5
```

The parameter RVAL is global and PARAM is the keyword used by the .STEP command when using a global parameter.

3) The following example steps the resistor model parameter R. This is another way of stepping a resistor from 30 to 50 ohms in steps of 5 ohms.

```
R1 1 2 RMOD 1
.MODEL RMOD RES(R=30)
.STEP RES RMOD(R) 30,50,5
```

.TEMP command

Purpose

The .TEMP command sets the temperature at which all analyses are done

General Form

```
.TEMP <temperature value>*
```

Examples

```
.TEMP 125
.TEMP 0 27 125
```

Comments The temperatures are in degrees Centigrade. If more than one temperature is given, then all analyses are performed for each temperature.

It is assumed that the model parameters were measured or derived at the nominal temperature, TNOM (27°C by default). See the .OPTIONS (analysis options) command for setting TNOM.

.TEMP behaves similarly to the list variant of the .STEP (parametric analysis) statement, with the stepped variable being the temperature.

.MODEL command

Purpose

The .MODEL command defines a set of device parameters that can be referenced by devices in the circuit.

General Form

```
.MODEL <model name>  <model type>
+ ([<parameter name> = <value> [tolerance specification]]*
+ [T_MEASURED=<value>] [[T_ABS=<value>] or  [T_REL_GLOBAL=<value>] or [T_REL_LOCAL=<value>]])
```

Examples

```
.MODEL RMAX RES (R=1.5 TC1=.02 TC2=.005)
.MODEL DNOM D (IS=1E-9)
.MODEL QDRIV NPN (IS=1E-7 BF=30)
.MODEL MLOAD NMOS(LEVEL=1 VTO=.7 CJ=.02pF)
.MODEL CMOD CAP (C=1 DEV 5%)
.MODEL DLOAD D (IS=1E-9 DEV .5% LOT 10%)
.MODEL RTRACK RES (R=1 DEV/GAUSS 1% LOT/UNIFORM 5%)
.MODEL QDR2 AKO:QDRIV NPN (BF=50 IKF=50m)
```

.MODEL command

Arguments and options

<model name>

The model name which is used to reference a particular model.

<reference model name>

The model types of the current model and the AKO (A Kind Of) reference model must be the same. The value of each parameter of the referenced model is used unless overridden by the current model, e.g., for QDR2 in the last example, the value of IS derives from QDRIV, but the values of BF and IKF come from the current definition. Parameter values or formulas are transferred, but not the tolerance specification. The referenced model can be in the main circuit file, accessed through a .INC command, or it can be in a library file;

<model type>

Must be one of the types outlined in the table that follows.

Devices can only reference models of a corresponding type; for example:

- A JFET can reference a model of types NJF or PJF, but not of type NPN.
- There can be more than one model of the same type in a circuit, although they must have different names.

Following the <model type> is a list of parameter values enclosed by parentheses. None, any, or all of the parameters can be assigned values. Default values are used for all unassigned parameters. The lists of parameter names, meanings, and default values are found in the individual device descriptions.

.MODEL command

Model type	Instance name	Type of device
CAP	Cname	capacitor
CORE	Kname	nonlinear, magnetic core (transformer)
D	Dname	diode
DINPUT	Nname	digital input device (receive from digital)
DOUTPUT	Oname	digital output device (transmit to digital)
GASFET	Bname	N-channel GaAs MESFET
IND	Lname	Inductor
ISWITCH	Wname	current-controlled switch
LPNP	Qname	lateral PNP bipolar transistor
NIGBT	Zname	N-channel insulated gate bipolar transistor (IGBT)
NJF	Jname	N-channel junction FET
NMOS	Mname	N-channel MOSFET
NPN	Qname	NPN bipolar transistor
PJF	Jname	P-channel junction FET

.MODEL command

Model type	Instance name	Type of device
PMOS	Mname	P-channel MOSFET
PNP	Qname	PNP bipolar transistor
RES	Rname	resistor
TRN	Tname	lossy transmission line
UADC	Uname	multi-bit analog-to-digital converter
UDAC	Uname	multi-bit digital-to-analog converter
UDLY	Uname	digital delay line
UEFF	Uname	edge-triggered flip-flop
UGATE	Uname	standard gate
UGFF	Uname	gated flip-flop
UIO	Uname	digital I/O model
UTGATE	Uname	tristate gate
VSWITCH	Sname	voltage-controlled switch

.DISTRIBUTION command

Purpose

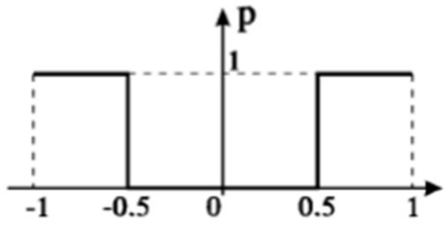
The .DISTRIBUTION command defines a user distribution for tolerances, and is only used with Monte Carlo and sensitivity/worst-case analyses. The curve described by a .DISTRIBUTION command controls the relative probability distribution of random numbers generated by PSpice to calculate model parameter deviations.

General Form

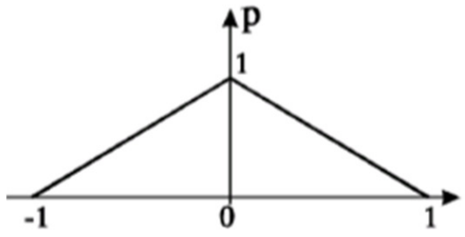
DISTRIBUTION <name> (<deviation> <probability>)*

Examples

.DISTRIBUTION bi_modal (-1,1) (-.5,1) (-.5,0) (.5,0) (.5,1) (1,1)



.DISTRIBUTION triangular (-1,0) (0,1) (1,0)



.DISTRIBUTION command

Arguments and options

(<deviation> <probability>)

Defines the distribution curve by pairs, or corner points, in a piecewise linear fashion. You can specify up to 100 value pairs.

<deviation>

Must be in the range (-1,+1), which matches the range of the random number generator.

No <deviation> can be less than the previous <deviation> in the list, although it can repeat the previous value.

<probability>

Represents a relative probability, and must be positive or zero.

.SUBCKT and .ENDS commands

Purpose

The .SUBCKT command/statement starts the subcircuit definition by specifying its name, the number and order of its terminals, and the names and default parameters that control its behavior. Subcircuits are instantiated using X (Subcircuit instantiation) devices. The .ENDS command marks the end of a subcircuit definition

General Form

```
.SUBCKT <subckt_name> [node]*
+ [OPTIONAL: < <interface node> = <default value> >*]
+ [PARAMS: < <param_name> = <value> >* ]
        ...
.ENDS
```

.SUBCKT and .ENDS commands

Examples

```
.SUBCKT OPAMP 1 2 101 102 17
...
.ENDS
.SUBCKT FILTER INPUT, OUTPUT PARAMS: CENTER=100kHz,  BANDWIDTH=10kHz
...
.ENDS
.SUBCKT PLD IN1 IN2 IN3 OUT1  PARAMS: MNTYMXDLY=0 IO_LEVEL=0
...
.ENDS
.SUBCKT 74LS00 A B Y
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
...
.ENDS
```

.SUBCKT and .ENDS commands

Arguments and options

<subckt_name>

The subckt_name is used by an X (Subcircuit Instantiation) device to reference the subcircuit.

[node]*

An optional list of nodes (pins). This is optional because it is possible to specify a subcircuit that has no interface nodes.

OPTIONAL:

Allows specification of one or more optional nodes (pins) in the subcircuit definition.

Comments

The subcircuit definition ends with a .ENDS command. All of the netlist between .SUBCKT and .ENDS is included in the definition. Whenever the subcircuit is used by an X (Subcircuit Instantiation) device, all of the netlist in the definition replaces the X device.

There must be the same number of nodes in the subcircuit calling statements as in its definition. When the subcircuit is called, the actual nodes (the ones in the calling statement) replace the argument nodes (the ones in the defining statement).

.SUBCKT and .ENDS commands

Comments

The optional nodes are stated as pairs consisting of an interface node and its default value. If an optional node is not specified in an X device, its default value is used inside the subcircuit; otherwise, the value specified in the definition is used.

This feature is particularly useful when specifying power supply nodes, because the same nodes are normally used in every device. This makes the subcircuits easier to use because the same two nodes do not have to be specified in each subcircuit statement. This method is used in the libraries provided with the Digital Simulation feature.

Subcircuits can be nested. That is, an X device can appear between .SUBCKT and .ENDS commands. However, subcircuit definitions cannot be nested. That is, a .SUBCKT statement cannot appear in the statements between a .SUBCKT and a .ENDS.

Subcircuit definitions should contain only device instantiations (statements without a leading period) and possibly these statements: .IC (initial bias point condition); .NODESET (set approximate node);

.SUBCKT and .ENDS commands

Comments

Models, parameters, and functions defined within a subcircuit definition are available only within the subcircuit definition in which they appear. Also, if a .MODEL, .PARAM, or a .FUNC statement appears in the main circuit, it is available in the main circuit and all subcircuits.

Node, device, and model names are local to the subcircuit in which they are defined. It is acceptable to use a name in a subcircuit which has already been used in the main circuit. When the subcircuit is expanded, all its names are prefixed using the subcircuit instance name: for example, Q13 becomes X3.Q13 and node 5 becomes X3.5 after expansion. After expansion all names are unique. The only exception is the use of global node names that are not expanded.

The keyword PARAMS: passes values into subcircuits as arguments and uses them in expressions inside the subcircuit. The keyword TEXT: passes text values into subcircuits as arguments and uses them as expressions inside the subcircuit.

.SUBCKT and .ENDS commands

Usage examples

1)In the example of the 74LS00 subcircuit, the following subcircuit reference uses the default power supply nodes $G_DPWR and $G_DGND:

```
X1 IN1 IN2 OUT 74LS00
```

2) To specify your own power supply nodes MYPOWER and MYGROUND, use the following subcircuit instantiation:

```
X2 IN1 IN2 OUT MYPOWER MYGROUND 74LS00
```

3) If wanted, one optional node in the subcircuit instantiation can be provided. In the

following subcircuit instantiation, the default $G_DGND would be used:

```
X3 IN1 IN2 OUT MYPOWER 74LS00
```

4) However, to specify values beyond the first optional node, all nodes previous to that node must be specified. For example, to specify your own ground node, the default power node before it must be explicitly stated:

```
X4 IN1 IN2 OUT $G_DPWR MYGROUND 74LS00
```

.SUBCKT and .ENDS commands

Usage examples

5) .SUBCKT OPAMP 1 2 10 11 14

```
...
.ENDS
X1 2 6 50 51 7 OPAMP
X2 0 13 50 51 27 OPAMP
```

In this example the subcircuit OPAMP is referenced twice.

In the first instance, X1, the nodes 1, 2, 10, 11 and 14 of the subcircuit are connected to nodes 2, 6, 50, 51 şi 7 from the circuit.

In the second instance, X2, the nodes 1, 2, 10, 11 and 14 of the subcircuit are connected to the nodes 0, 13, 50, 51 and 27 from the circuit..

Example 6:

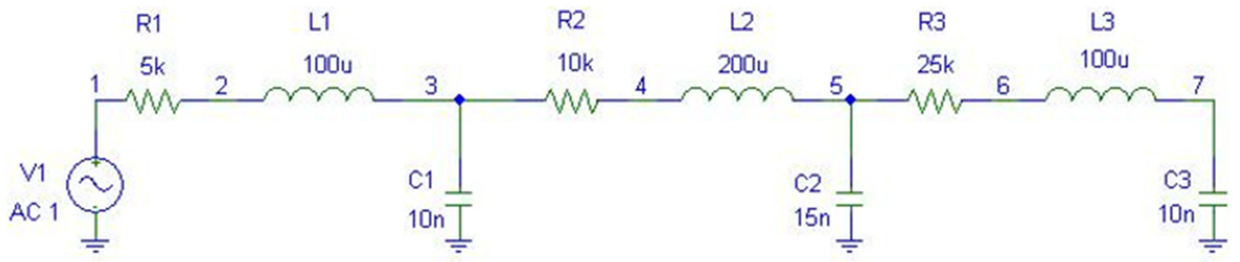
```
 .SUBCKT RC_cell 1 3 PARAMS: RVAL=3k
R1 1 2 {RVAL}
R2 2 3 {2*RVAL}
C1 1 0 100N
C2 2 0 50N
.ENDS
```

Applications

Application 1

Activities: 1). Describe into the SPICE circuit file (.cir) the circuit shown in the below figure. The circuit will be seen as three serial connected RLC cell. To do this, describe an RLC cell as subcircuit having local parameters RVAL, LVAL, CVAL with default values of 5k, 100u and 10n respectively. This subcircuit will be referenced (instantiated) 3 times in the circuit (X1, X2 and X3). For instances X2 and X3 effective values of parameters RVAL, LVAL and CVAL shall be specified.

2) Perform an AC analysis for the described circuit in the frequency interval (10Hz, 100MHz) and show in Probe the transfer functions from the input to the outputs of the three instances X2, X2 and X3.



Applications

Application 2

1) Describe into the SPICE circuit file (.cir) the following circuit which contains an Operational Amplifier uA741

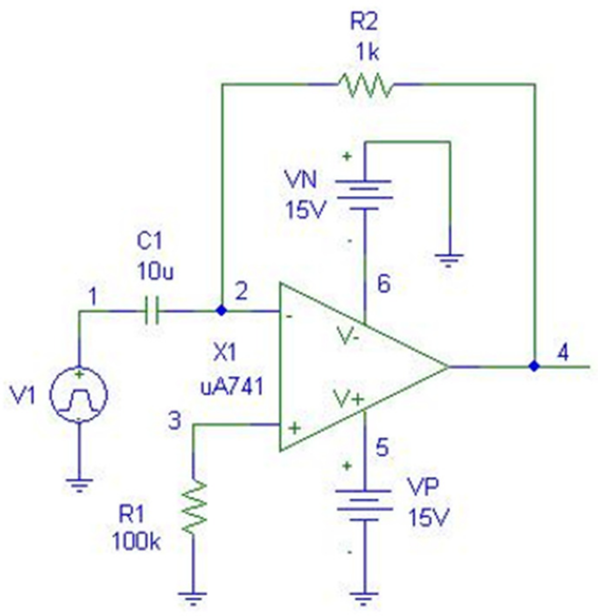
Note: The OA uA741 is described in the library as as subcircuit having the subckt_name uA741 and the order of the interface nodes is IN+, IN-, V+, V-, OUT.

In the circuit description you must edit an instance X1 of the subcircuit uA741.

The voltage source V1 provides the waveform shown in the next slide.

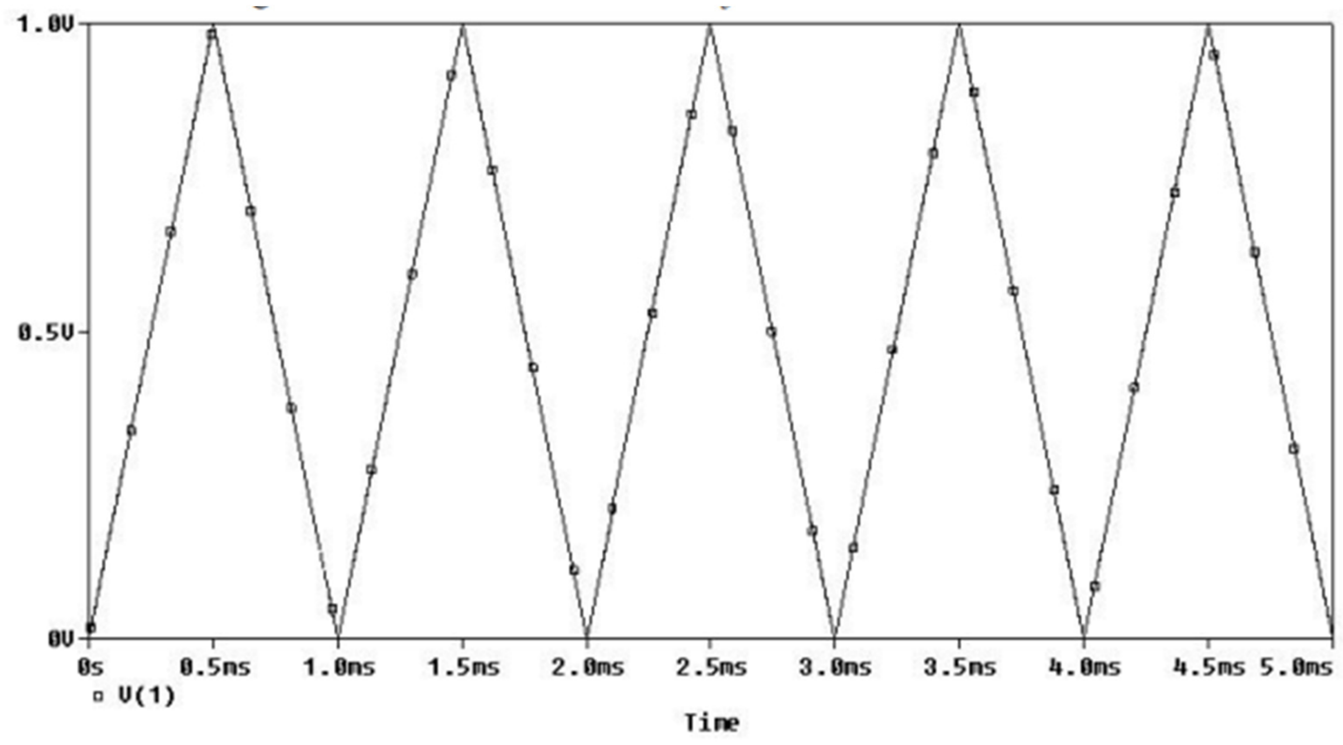
2) Perform an transient analysis of the circuit considering the following transient parameters:

.TRAN 1u 5m 0 100u



Applications

The waveform generated by independent voltage source V1


1.0V
0.5V
0V
0s
0.5ms
1.0ms
1.5ms
2.0ms
2.5ms
3.0ms
3.5ms
4.0ms
4.5ms
5.0ms
□ V(1)
Time